Interpreting American Sign Language with Convolutional Neural Networks

Hosung Hwang
hhwang4@ncsu.edu

Alex Carruth
agcarrut@ncsu.edu

Utsab Ray
uray@ncsu.edu

I. Motivation

According to WHO [1] there are 466 million people in the world with disabling hearing loss. A big percentage of these people rely on sign language to communicate with their family, friends and other people around them. With an increase in the number of people with disabling hear loss every year, it is important for machines to understand sign language, to make the lives of these people easier and to encourage a more inclusive society.

Previous work done in this area has used Convolutional Neural Networks (CNNs) to analyze the database. A recent model created by Ameen and Vadera analyzed the American Sign Language dataset to a precision of 82% [2]. While most previous networks for American Sign Language have used image preprocessing, few have strictly studied the effects of using different image preprocessing techniques have on the model. Further areas of research in image preprocessing techniques can be emphasized by examining the effect of current individual methods on improving a CNN model.

II. Data

For this project, we used a collection of publicly available images of alphabets from the American Sign Language found on Kaggle. The training data set contains 87,000 images of human hands formed in the shapes of the ASL alphabet, each with dimensions of 200x200 pixels. This data set was divided into 29 separate categories, 26 of which are used to test the letters A-Z, and one class each to test ”Space”, ”Delete”, and ”Nothing”.

The standard training data set that was obtained from Kaggle was divided so that enough data was available for the validation set as well. As per the standard ratio learned in class, the training data was split so that 80% was allotted for the training data and the remaining 20% was allocated for the validation data set.

The test data set consisted of 28 images with the same classes as the training set except for the DELETE category, as well as another set of 56 images synthesized from the standardized 28 image data set obtained from Kaggle. The synthesized data set consisted of images that changed the brightness and zoom levels of the original 28 images in the standardized test data set. Our team added this synthesized set in order to expand the set to obtain more realistic analysis on how the model might work on images in the real world and to apply data synthesis techniques we had learned in the course. Figure 1.a and Figure 1.b show images from the test dataset augmented with zoom and brightness, respectively. Another test dataset was also made for every letter in the ASL alphabet with the hands of each group member. The images in Figure 2 show an example of what the pictures of the hands of each group member looked like. Their were certain key elements missing from this dataset that were found in the given test dataset such as the original size of the images (200x200px) and the blue outline of the images in the dataset.

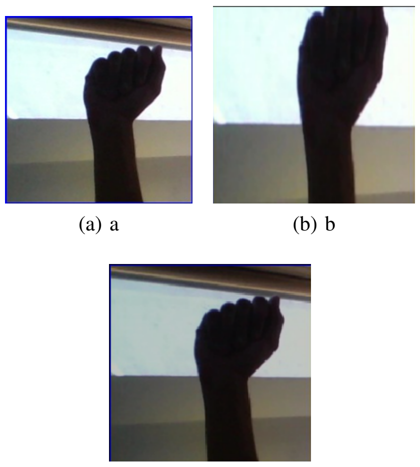
(a) a (b) b (c) c

Fig. 1: Image from the ASL test dataset for the letter ’A’: Figure a is for the original image; Figure b is for the image augmented with zoom; and, Figure c is for the image augmented with brightness.

III. Methodology

For computer vision models, the neural network of choice is often the Convolutional Neural Network. Image preprocessing can have a great effect on the success of these networks. Thus, we chose to compare a baseline Convolutional Neural Network with no data preprocessing to ones with image preprocessing applied.

NN with no data preprocessing as the baseline model for this project. CNN’s are commonly used in image processing because they are able to take images with high dimensionality and funnel them into a reduced form without losing much of

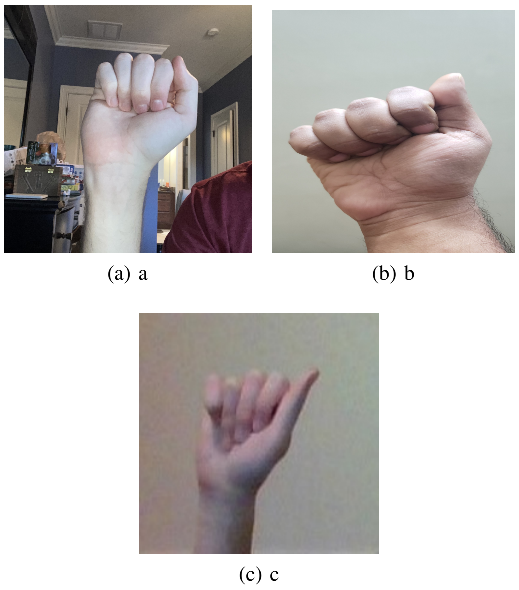

Fig. 2: Images from the self-made dataset for the letter ’A’: Figure a is a picture of Alex’s hand, Figure b is a picture of Utsab’s hand, and Figure c is a picture of Hosung’s hand.

the information [3]. In a CNN, a sliding window matrix works through the data as a convolutional filter to analyze an object in the image.

B) Preprocessing Models:

There were three models which we compared to the baseline CNN which each utilized different image preprocessing techniques. In related work conducted by Lopes *et al.*, rotation correction, image cropping, down-sampling and intensity normalization were used to create a more efficient fake face detection CNN [4]. Our model implemented this same preprocessing scheme for the American Sign Language dataset to see its significance on improving the CNN model. Three CNN models with different preprocessing schemes based on those used by Lopes *et al.* were constructed. The first was a spatial normalization model which used image cropping. Originally, image modifications which rotated images was used as part of the spatial normalization scheme but it was found to reduce the accuracy of all predictions significantly so the results were not used in this paper. The next model was a intensity normalization model which normalized the pixels for the images of our dataset. The third was a model which implemented both spatial and intensity normalization.

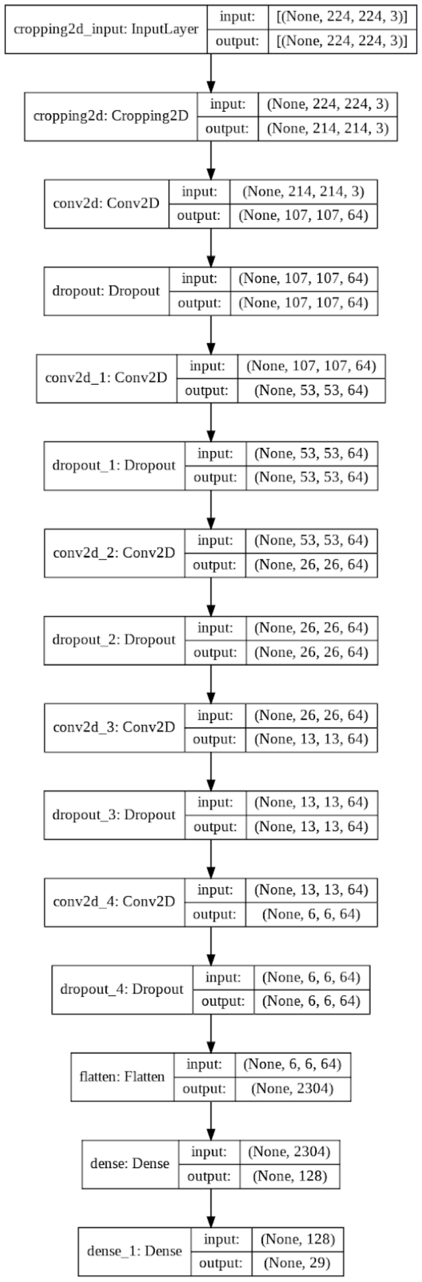
cropping2d_input: InputLayer | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]
cropping2d: Cropping2D | input: (None, 224, 224, 3) | output: (None, 214, 214, 3)
conv2d: Conv2D | input: (None, 214, 214, 3) | output: (None, 107, 107, 64)
dropout: Dropout | input: (None, 107, 107, 64) | output: (None, 107, 107, 64)
conv2d_1: Conv2D | input: (None, 107, 107, 64) | output: (None, 53, 53, 64)
dropout_1: Dropout | input: (None, 53, 53, 64) | output: (None, 53, 53, 64)
conv2d_2: Conv2D | input: (None, 53, 53, 64) | output: (None, 26, 26, 64)
dropout_2: Dropout | input: (None, 26, 26, 64) | output: (None, 26, 26, 64)
conv2d_3: Conv2D | input: (None, 26, 26, 64) | output: (None, 13, 13, 64)
dropout_3: Dropout | input: (None, 13, 13, 64) | output: (None, 13, 13, 64)
conv2d_4: Conv2D | input: (None, 13, 13, 64) | output: (None, 6, 6, 64)
dropout_4: Dropout | input: (None, 6, 6, 64) | output: (None, 6, 6, 64)
flatten: Flatten | input: (None, 6, 6, 64) | output: (None, 2304)
dense: Dense | input: (None, 2304) | output: (None, 128)
dense_1: Dense | input: (None, 128) | output: (None, 29)

Fig. 3: 1D CNN Network Architecture

Table 1: Validation Loss and Accuracy for Initial CNN Model

Epoch	Validation Loss	validation Accuracy
1	1.0366	0.5659
2	1.0194	0.5469
3	0.9995	0.6096
4	1.1280	0.5455
5	1.1094	0.5896
6	0.9969	0.6493
7	1.0656	0.6189
8	1.0941	0.5840
9	1.2100	0.5661
10	1.1194	0.6193
11	1.0436	0.6526
12	1.1191	0.6195
13	1.0460	0.6362
14	1.1372	0.5990
15	1.1197	0.6028

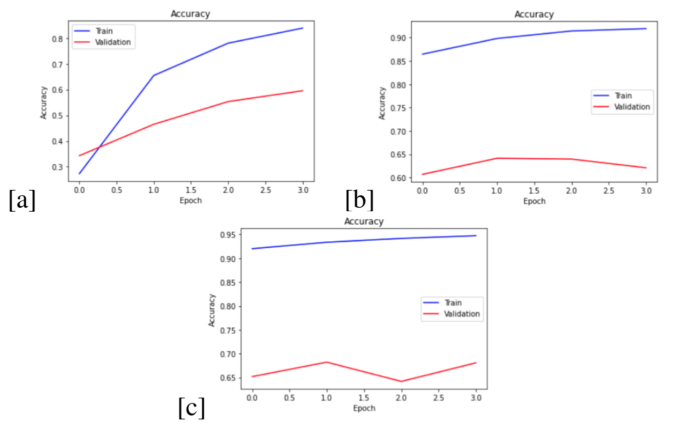

Fig. 4: Accuracy plots for three different optimizers: Figure a is for the model with SGD; Figure b is for the model with RMSProp; and, Figure c is for the model with ADAM.

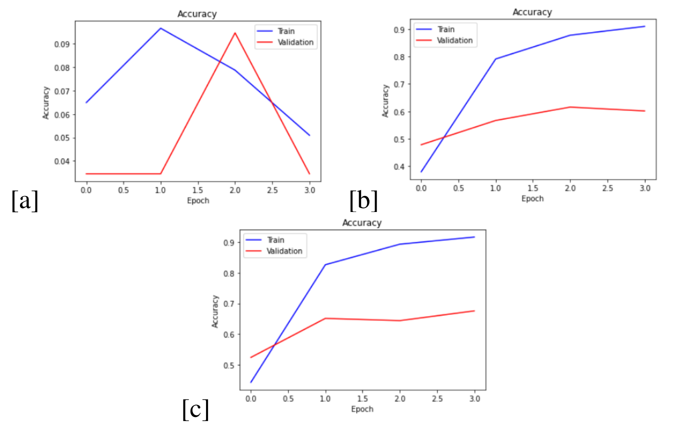

Fig. 5: Accuracy plots for three different activation functions: Figure a is for the model with relu; Figure b is for the model with sigmoid; and, Figure c is for the model with softmax.

IV. Model Training and Selection

A. Model Training

The training data set was split into 80% training and 20% validation. Additionally, the images in the training data set underwent pre-processing in three different ways: image cropping, image normalization, and image cropping+normalization.

The CNN model that we constructed to train and fit our model had five Conv2D layers, each with strides of 2, kernel size of 2, and 'relu' activation. Dropout layers (each with rates of 0.2) were used to complement each Conv2D layer utilized, and a flatten layer was used to flatten the input data. Two dense layers with differing units and activation function types were used: the units were 128 and 29, and the activation functions used were 'relu' and 'softmax'.

B. Model Selection

Figure 4 shows the experiments we conducted by choosing three different optimizers and the accuracy values that we got. For model selection purposes, we decide to go till 3 epochs and check the accuracies. As we can see from the plots, the model with the ADAM optimizer gave us the most promising training and validation accuracy, and that is why we went forward with that. Figure 4 shows the experiments we conducted by choosing three different activation functions and the accuracy values that we got. As we can see from the plots, softmax and sigmoid have similar training accuracies. However softmax edges out sigmoid in terms of validation accuracy. That is why we went with softmax

V. Evaluation and Results

To test and evaluate the efficacy of our model, we will be using the publicly available test data set along with the same images with modified zoom and brightness levels.

In Figure 7 we plot accuracy and loss values for the 4 models we implemented; one with no preprocessing, one with image cropping, one with intensity normalization and one with both intensity normalization and image cropping. From the plots we can see that the model with both intensity normalization and image cropping has the highest training and validation accuracy.

Figure 6 shows a chart where we've plotted accuracy and F-1 scores for models with different preprocessing schemes. We record the accuracy and F-1 score of each model twice. Once for the ASL test dataset, and once for the ASL test dataset augmented with zoom and brightness. From the chart we can see that for the ASL test dataset, the model with image cropping and the model with both image cropping and image normalization work best. However for the augmented dataset, the baseline model with no preprocessing turned out to have the best accuracy and F-1 scores

While not officially noted in our report, our team also tested our models out using test data garnered amongst ourselves: that is to say, we took photos of our own hands in the shape of the ALS alphabet letters and tested our model against them. While a novel idea, the accuracy rates and F-1 scores of our models when making predictions on our own data set was never more than 5%, substantially lower than when predicting the hand images in the standard data set and its synthesized derivatives. The low accuracy rates and F-1 scores measured was possibly due to the distortion that some of these hand-crafted images had when we tried to conform them to the 200x200 pixel sizes, as well as the varying levels of lighting that were present in the data sets.

VI. Discussion and Conclusion

Through analysis of the CNN model our team constructed, it was deemed that our CNN models with image cropping and image cropping+normalization pre-processing methods had the best F-1 scores and accuracies when using the standard ASL test dataset (unsynthesized). In the model's use of both pre-processing techniques, the accuracy and F-1 score was 100% and 1.00, respectively. However, when using test datasets synthesized from the standard ASL test dataset, the baseline CNN model with no image pre-processing produced

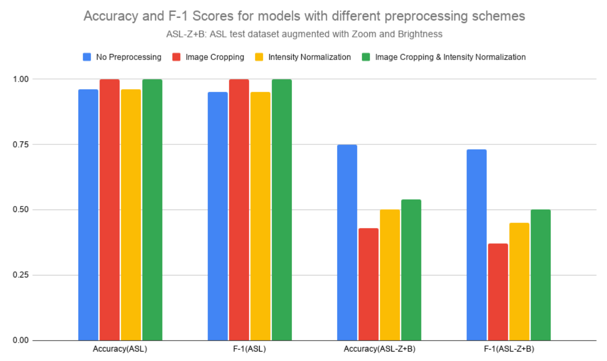

Fig. 6: Accuracy and F-1 scores for models with different preprocessing schemes

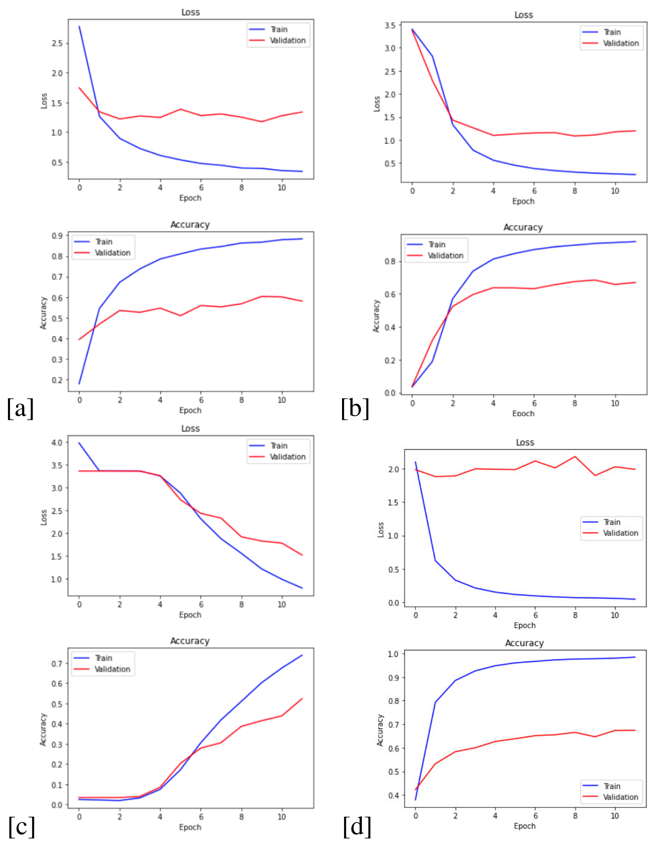

Fig. 7: Accuracy and loss plots for four different models: Figure a is for the model with no preprocessing; Figure b is for the model with image cropping; Figure c is for the model with intensity normalization; and, Figure d is for the model with both intensity normalization and image cropping.

the best accuracy and F-1 score (75% and 0.73, respectively). Thus, our model was significantly more accurate in identifying ASL hand images than had guessing been used. Compared to the accuracy rate garnered purely from randomly guessing (3.57%), our model had accuracy rates that were magnitudes greater.

The results makes sense when analyzing the purpose of using each of the two pre-processing technqiues. While image cropping improves the overall image composition by extracting out extraneous regions that hamper identification of the object being analyzed (in this case, the outline of the hand spelling out an ASL alphabet letter), intensity normalization normalizes the pixels for the images of our dataset, bringing the overall image into a range that facilitates object recognition. When combining these two pre-processing techniques, there seemed to be an advantageous gain in the CNN model's ability to accurately recognize hand images spelling out ASL alphabet letters.

While we had also augmented the test dataset by including pictures of our hands signing the alphabet, we did not add F-1 and accuracy scores for the same. This is because the F-1 and accuracy scores were extremely low for the pictures where we took of our own hands signing the alphabet. We hypothesize this is because of the absence of the blue outline in the images, which is present in the ASL training dataset. Along with that the lighting conditions and outline of our hands are not consistent with the training dataset. Thus it became very difficult for the model to infer the alphabet based on our pictures.

REFERENCES

[1] https://www.who.int/news-room/facts-in-pictures/detail/deafness
[2] Ameen, Salem, and Sunil Vadera. ”A convolutional neural network to classify American Sign Language fingerspelling from depth and colour images.” Expert Systems 34.3 (2017): e12197.
[3] https://medium.datadriveninvestor.com/why-are-convolutional-neural-networks-good-for-image-classification-146ec6e865e8
[4] Andr Teixeira Lopes, Edilson de Aguiar, Alberto F. De Souza, and Thiago Oliveira-Santos. 2017. Facial expression recognition with Convolutional Neural Networks. Pattern Recogn. 61, C (January 2017), 610–628. DOI:https://doi.org/10.1016/j.patcog.2016.07.026